Dokumentace úlohy DKA: determinizace konečného automatu v PHP 5 do IPP 2016/2017
Jméno a příjmení: Radek Mojžíš
Login: xmojzi07

1 Návrh

Úlohu jsem si dekomponoval na několik jednodušších podúloh, které jsem řešil zvlášť. Jako první načtu vstupní parametry, dále načtu vstupní řetězec a pomocí regulárních výrazů provedu syntaktickou kontrolu vstupu. Potom si vstup rozdělím a převedu řetězec na pětici množin konečného automatu. Množiny jsem se rozhodl reprezentovat pomocí polí, neboť PHP dokáže nad polemi provádět všechny základní množinové operace, jako je například průnik nebo sjednocení. Další řešená podúloha je odstranění epsilon přechodů a nakonec samotná determinizace konečného automatu.

2 Implementace

2.1 Načítání vstupních parametrů

Vstupní parametry skript načítá ve funkci `get_arguments` pomocí `getopts()`, také následně kontroluje, je-li kombinace přepínačů korektní a nastavuje globální proměnné jako například vstupní a výstupní soubor nebo flagy pro jednotlivé možnosti.

2.2 Načtení vstupu

Jak již jméno funkce napovídá, `load_input()` načítá vstup. Soubor, ze kterého se načítá je specifikován při načítání parametrů, nebo implicitně `STDIN`. Dále také dělá drobné úpravy v samotném vstupu, jako je například ignorování komentářů, nebo převádění na *lowercase*. Následně potom vrací celý vstup jako řetězec.

2.3 Parsování vstupu a syntaktická kontrola

Parsování probíhá ve funkci `parse_input()`. Její vstup je řeťězec s načteným automatem, na výstupu potom vrací ptětici polí konečného automatu. Parsování a syntaktická analýza probíhá s využitím regulárních výrazů. Nejprve hrubá struktura, neboli složené závorky, následně potom jemnější části jako například pravidla. V této části využívám několikrát i funkce `explode()`.

2.4 Sémantická kontrola

Kontrola sémantické správnosti vstupního automatu je řešena ve funkci `validate_fsm()`. Zde se kontrolují formality jako „Počáteční stav musí náležet množině stavů“. Zpravidla jde o volání `array_search()`.

2.5 Determinizace a odstraňování epsilon pravidel

V této části jsem opět přistoupil k jisté dekompozici problému. Vyčlenil jsem si funkci pro vytvoření epsilon uzávěru `epsilon_closure()` ale jinak jsem se nijak dramaticky neodchyloval od algoritmu popsaného ve skriptech předmětu IFJ. Jediná odlišnost spočívá v seřazení výstupních polí podle specifikací a v tom, že v IFJ není jasně uvedeno, že se nejprve musí odstranit případné epsilon přechody a až poté lze zahájit determinizaci.

2.6 Výstup

Nakonec je automat vypsán do zvoleného výstupního souboru, implicitně se vypisuje na `STDOUT`. O to se stará funkce `print_fsm()` která výstup formátuje přesně podle specifikací v zadání.

3 Závěr a zhodnocení

Celkově mě téma projektu nijak zvlášť nenadchlo. Bude to asi tím, že poslední dobou jediné, co píšeme jsou parsery textu v různých jazycích. Oceňuji z mého pohledu mimořádně dobře zadané specifikace. Za celou dobu psaní jsem si nepřišel,

že bych nevěděl, co dál dělat, a s výsledkem mého snažení jsem docela spokojený. Pro implementaci tohoto projektu mi nepřišlo příliš vhodné vytvářet nové objekty v php a tak jsem zůstal u těch „klasických“.